

Python: module cdms.CDMLParser

[cdms.CDMLParser](#)

[index](#)

Parse a CDML/XML file

Modules

[cdms.CDML](#)

[cdms.cdmsNode](#)

[re](#)

[string](#)

Classes

[cdms.cdxmllib.XMLParser](#)
[CDMLParser](#)

class ***CDMLParser***([cdms.cdxmllib.XMLParser](#))

Methods defined here:

__init__(self, verbose=0)

cndl_syntax_error(self, lineno, message)

close(self)

end_attr(self)

Now the value if any as stored in self.***content***

end_axis(self)

end_cdml(self)

end_component(self)

end_compoundAxis(self)

end_data(self)

end_dataset(self)

end_doclink(self)

end_domElem(self)

end_domain(self)

```
end_linear(self)

end_rectGrid(self)

end_variable(self)

end_xlink(self)

getCurrentNode(self)
    # Get the current parent node

getRoot(self)
    # Get the root node

handle_cdata(self, data)

handle_data(self, data)
    # Handle content
    # discard data which is just whitespace,
    # and strip other data

handle_proc(self, name, data)

handle_special(self, data)

handle_starttag(self, tag, method, attrs)

popCurrentNode(self)
    # Pop the current node off the stack

pushCurrentNode(self, node)
    # Push current node on the stack

start_attr(self, attrs)

start_axis(self, attrs)

start_cdml(self, attrs)
    #-----

start_component(self, attrs)

start_compoundAxis(self, attrs)
    #-----

start_data(self, attrs)
    #-----

start_dataset(self, attrs)

start_doclink(self, attrs)
```

```
start_domElem(self, attrs)

start_domain(self, attrs)
    #-----

start_linear(self, attrs)

start_rectGrid(self, attrs)

start_variable(self, attrs)

start_xlink(self, attrs)

unknown_charref(self, ref)

unknown_endtag(self, tag)

unknown_entityref(self, ref)

unknown_starttag(self, tag, attrs)
```

Methods inherited from [cdms.cdxmllib.XMLParser](#):

```
feed(self, data)
    # Interface -- feed some data to the parser. Call this as
    # often as you want, with as little or as much text as you
    # want (may include '\n'). (This just saves the text, all th
    # processing is done by goahead().)

finish_endtag(self, tag)
    # Internal -- finish processing of end tag

finish_starttag(self, tagname, attrdict, method)
    # Internal -- finish processing of start tag

getnamespace(self)
    # Interface -- return a dictionary of all namespaces currently

goahead(self, end)
    # Internal -- handle data as far as reasonable. May leave st
    # and data to be processed by a subsequent call. If 'end' is
    # true, force handling all data as if followed by EOF marker.

handle_charref(self, name)
    # Example -- handle character reference, no need to override

handle_comment(self, data)
    # Example -- handle comment, could be overridden

handle_doctype(self, tag, pubid, syslit, data)
    # Overridable -- handle DOCTYPE
```

```

handle_endtag(self, tag, method)
    # Overridable -- handle end tag

handle_xml(self, encoding, standalone)
    # Overridable -- handle xml processing instruction

parse_attributes(self, tag, i, j)
    # Internal -- parse attributes between i and j

parse_cdata(self, i)
    # Internal -- handle CDATA tag, return length or -1 if not terminated

parse_comment(self, i)
    # Internal -- parse comment, return length or -1 if not terminated

parse_doctype(self, res)
    # Internal -- handle DOCTYPE tag, return length or -1 if not terminated

parse_endtag(self, i)
    # Internal -- parse endtag

parse_proc(self, i)
    # Internal -- handle a processing instruction tag

parse_starttag(self, i)
    # Internal -- handle starttag, return length or -1 if not terminated

reset(self)
    # Interface -- reset this instance. Loses all unprocessed data

setliteral(self, *args)
    # For derived classes only -- enter literal mode (CDATA)

setnomoretags(self)
    # For derived classes only -- enter literal mode (CDATA) till end

syntax_error(self, message)
    # Example -- handle relatively harmless syntax errors, could be fatal

translate_references(self, data, all=1)
    # Interface -- translate references

```

Data and other attributes inherited from [cdms.cdxmllib.XMLParser](#):

```

attributes = {}

elements = {}

entitydefs = {'amp': '&#38;', 'apos': '&#39;', 'gt': '&#62;', 'lt': '&#60;', 'quot': '&#34;'}

```

Data

InvalidAttribute = 'Invalid attribute'